Shared Documents - Design Doc

**Status:** Draft

**Created:** Nov 03, 2022

**Last Updated:** Nov 08, 2022

**Authors:** Ana Levit, Tal Ben-Zur, Lior Mathan

**Additional Links**: [Simple Project Plan](https://docs.google.com/document/d/1R68EAEgets8udkKx6rzCO3_GqXwimrWRABOFQud-BGg/edit?usp=sharing) [Shared Doc req.pdf](https://drive.google.com/file/d/1fKQkn8He63isOvf0L3mgw0ppgdt_lz3o/view?usp=share_link)

# Background

There are tons of text editors out there, some more well known and some less.

Most of them offer editing documents locally, require saving so as not to lose the work done, and in order to share them with others - they must be sent in some way.

In this document we will present the Shared Documents application design.

This application is a web-based editor that allows users to organize files (documents) into folders and includes editing, creating, sharing and removing functionalities.

The application allows several collaborators to update a single document simultaneously in an online editor.

More documentation:

* [User Stories - Shared docs application](https://docs.google.com/document/u/0/d/10sVY9hlJuZZtgE6CdJgkhnFz6ztOMp9cKYMnZis94pc/edit)
* [User Journey - Shared Documents Application](https://docs.google.com/document/u/0/d/1hELpy8nvjB4BWiHDN1y4qLt1ZWn1gZCTYO8FsL9o35k/edit)
* [Application Requirements](https://docs.google.com/spreadsheets/u/0/d/1L22Ft7M1RXdPN9_a0fJ4XLtwsbXXeoeBRQt2DAjHAV4/edit)

# Objective

The system should support the following key requirements:

1. Online editing and formatting of a document, of several collaborating users simultaneously.

2. Share documents and assign different permissions (owner, read-only, edit).

3. Organize files (documents) into folders.

# 

# Overview

* Basic classes:
  + Document class - holds the document's metadata, authorized and active users, and the document’s content.
  + User class - holds user data and all documents linked to the user.
  + Editor class - allows updating and saving documents.
* All document editing operations are initiated in the editor class.
* The application’s API classes are: AuthenticationController, UserController and DocumentController and they control the following services respectively: AuthenticationService, UserService, DocumentService.
* The Document class holds the file’s metadata (in Metadata object) which contains the creation date, the parent directory, file’s URL, and filename.
* All users DB operations are carried out in UserRepository class.
* All document DB operations are carried out in DocumentRepository class.
* Permission is an enum which consists of the values: ADMIN, EDITOR, VIEWER.

# Alternative Solutions

There are two approaches to update a shared file that we will focus on

1. Updating in real time - the update operation will be carried out in real time, this means that every change that a user with editing permissions will do (typing in / deleting / modifying text) will automatically update for all collaborators.

2. Delayed updating - the operation will be carried out every X seconds, this means that every change that a user with editing permissions will do (typing in / deleting / modifying text) will update for all collaborators only every that period time.

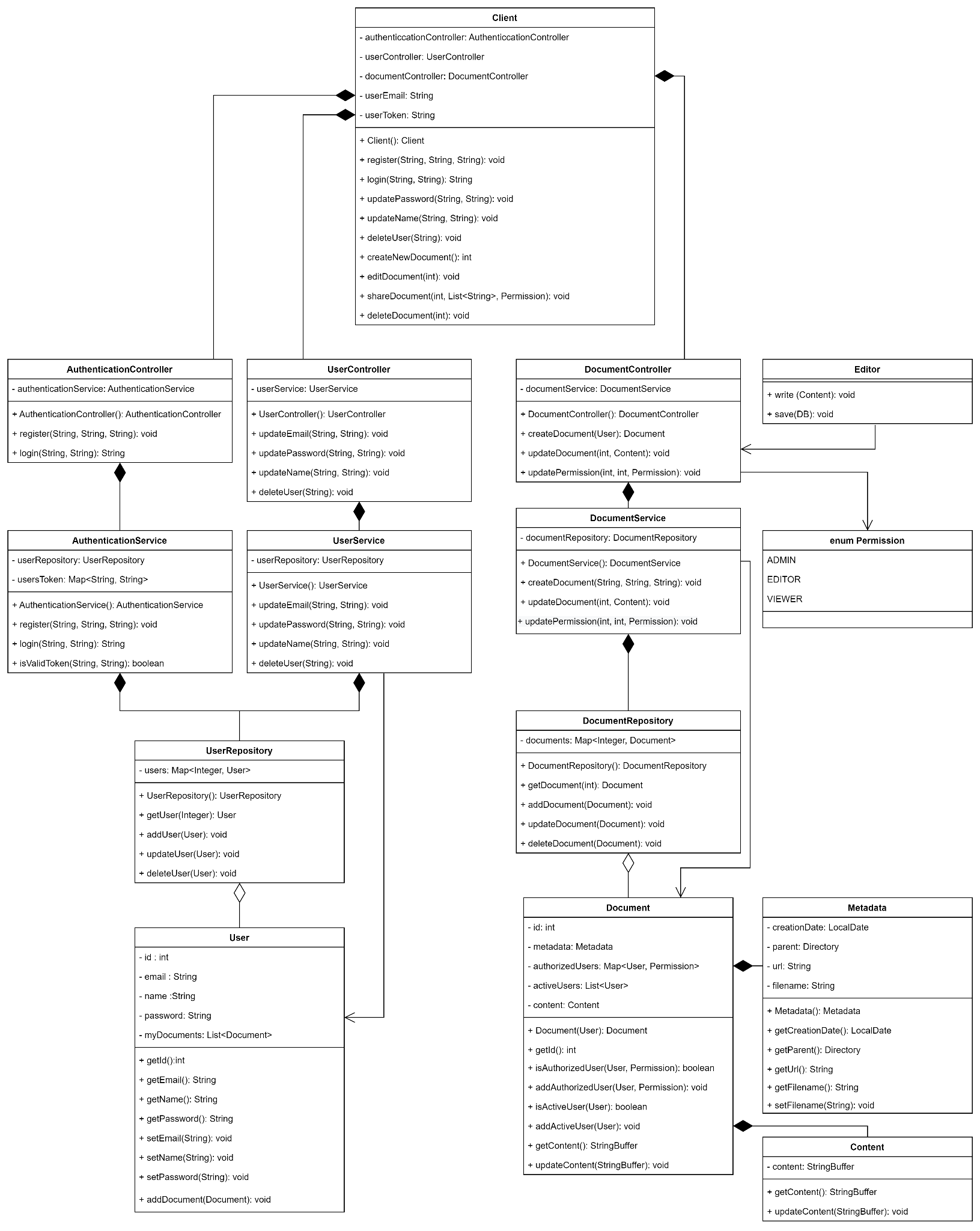
|  | Updating in real time | Delayed updating |
| --- | --- | --- |
| **Performing actions** | The document updates immediately after a change is made. | The document updates every X seconds, no matter when a change was made. |
| **Runtime efficiency** | A notification is made only when an update occurs. | There may be a lot of unnecessary notifications. |
| **Overlap between updates** | There is minimum overlap between updates, everything is managed in a queue | A situation of overlap between updates can arise. Complicated conflicts may arise. |

# The Bottom Line:

In the delayed updating, updates are more unsafe and cause lots of unnecessary notifications, in the immediate updating, the operation is carried out immediately and faster, therefore causing less conflicts. we chose to use an immediate updating.

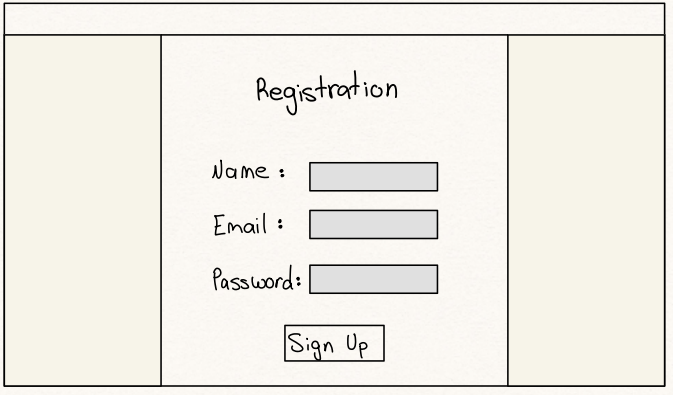
# Detailed Design

## **UML Diagram (class diagram)**

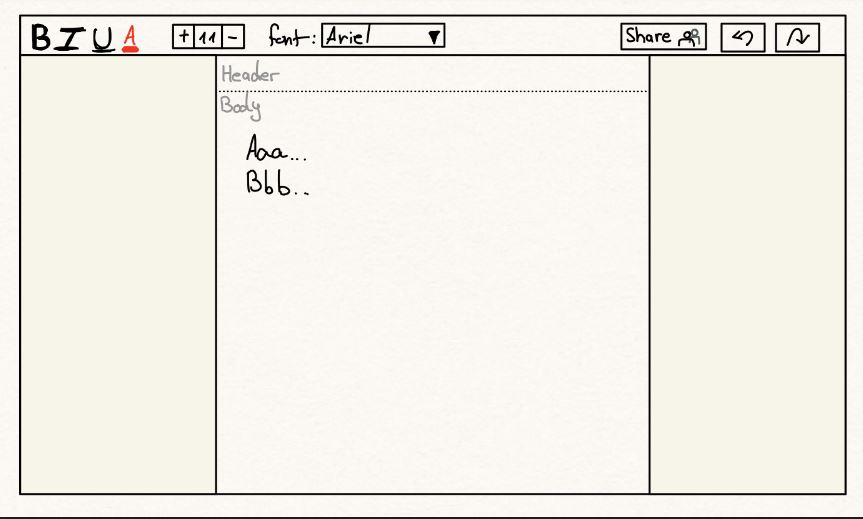
[UML class diagram](https://drive.google.com/file/d/1sTfzeHjk-Hjmmlz6dSod6aupiPCvJkgw/view?usp=sharing)

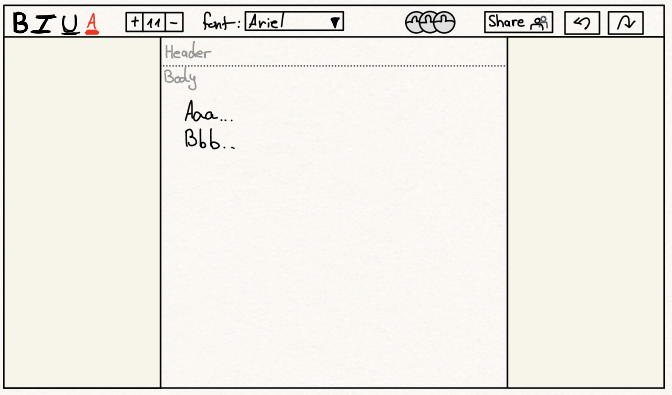
## **Sketch**

Registration page:



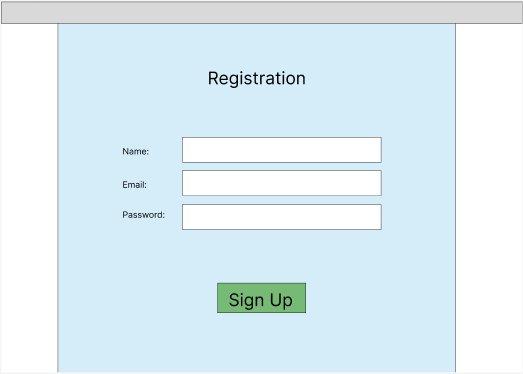
Edit permission:

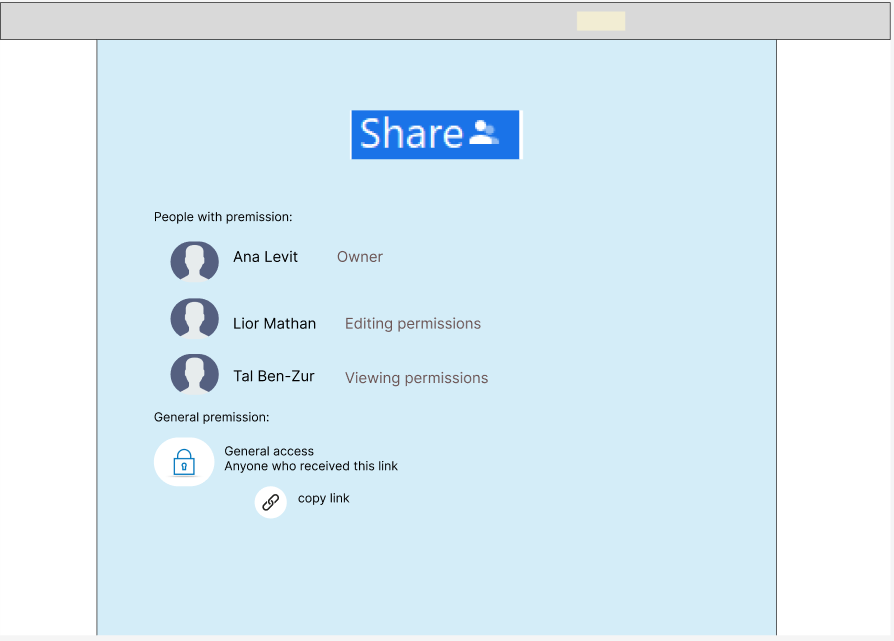


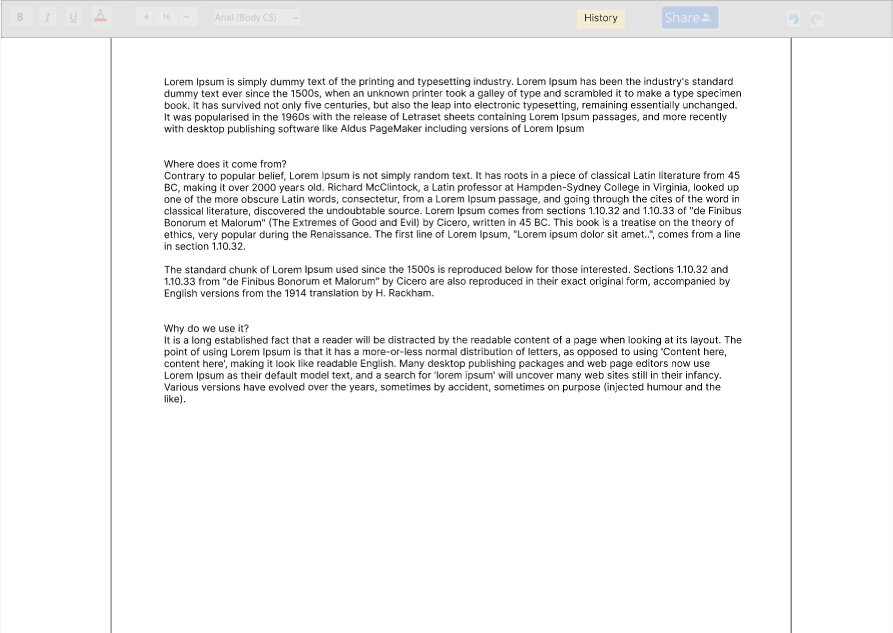
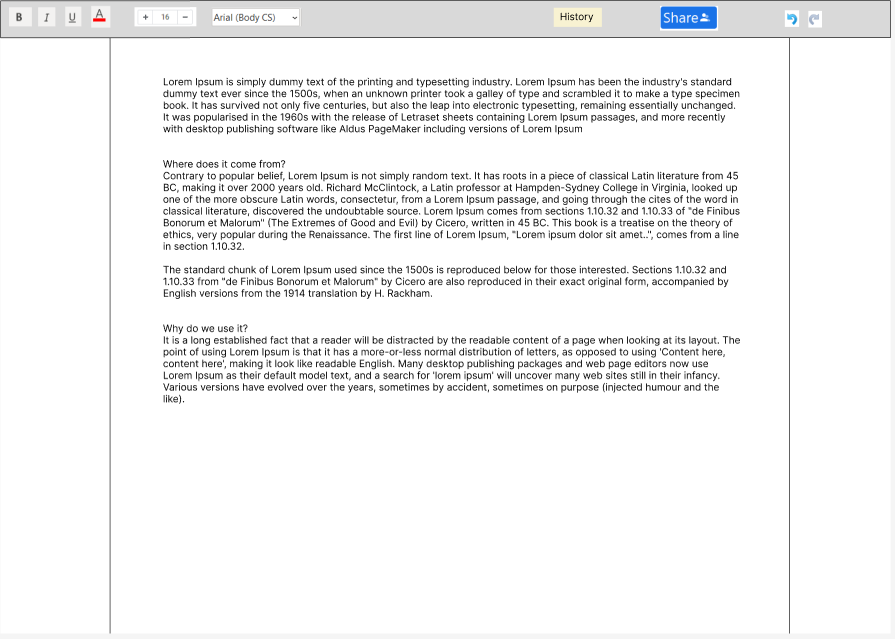
Edit simultaneously: can see all active collaborators.

## 

## **Mockup**

Registration page: 

Share page: 

View permission:edit permission:

Edit simultaneously: can see who else edit.